Лабораторна робота №4

Тема: Дослідження методів неконтрольованого навчання

Завдання 1: Кластерізація данних за допомогою методу к-середніх

*Лістинг програми:*

#Task1  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
from sklearn import metrics  
  
X = np.loadtxt('data\_clustering.txt', delimiter=',')  
num\_clusters = 5  
  
#Chart  
  
plt.figure()  
plt.scatter(X[:,0], X[:,1], marker='o', facecolors='none',  
 edgecolors='black', s=80)  
x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1  
y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1  
  
plt.title('Input Data')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
#chart add to the report  
  
kmeans = KMeans(init='k-means++', n\_clusters=num\_clusters, n\_init=10)  
kmeans.fit(X)  
  
step\_size=0.01  
  
x\_min, x\_max = X[:, 0].min() - 1, X[:,0].max() + 1  
y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1  
x\_vals, y\_vals = np.meshgrid(np.arange(x\_min, x\_max, step\_size),  
 np.arange(y\_min, y\_max, step\_size))  
  
output = kmeans.predict(np.c\_[x\_vals.ravel(), y\_vals.ravel()])  
output = output.reshape(x\_vals.shape)  
plt.figure()  
plt.clf()  
plt.imshow(  
 output,  
 interpolation='nearest',  
 extent=(x\_vals.min(), x\_vals.max(), y\_vals.min(), y\_vals.max()),  
 cmap=plt.cm.Paired, aspect='auto', origin='lower')  
  
plt.scatter(X[:, 0], X[:, 1],  
 marker='o', facecolors='none', edgecolors='black', s=80)

cluster\_centers = kmeans.cluster\_centers\_  
plt.scatter(cluster\_centers[:,0], cluster\_centers[:, 1],  
 marker='o', s=210, linewidths=4, color='black',  
 zorder=12, facecolors='black')  
  
x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1  
y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1  
  
plt.title("Claster Edges")  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
  
plt.xticks(())  
plt.yticks(())  
plt.show()  
#Chart Add to report

Після виконання програми отримали наступні два графіки (рис.1) Ям можна бачити за допомогою обраного методу визначення подібності вдалося створити пять кластерів зосередження даних. Визначивши центральні точки підгруп всередені набору даних. Також можна помітити, що деяка кількість точок відноситься до погранічних ділянок й наближені одразу до декількох кластерів.

Chart, scatter chart, qr code

Description automatically generatedA picture containing chart

Description automatically generated

Рисунок 1. Отримані графіки К-середніх

Завдання 2: Кластерізація К-середніх для нобору даних Iris

Використаємо для кластерізації дані з наборів (набір даних ірисів), які вже використовували в попередніх роботах

*Лістинг програми:*

#TASK 2  
import numpy as np  
import sklearn  
from sklearn.svm import SVC  
from sklearn.metrics import pairwise\_distances\_argmin  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
from sklearn import metrics  
from sklearn.datasets import load\_iris  
  
iris = load\_iris()  
X = iris['data']  
y = iris['target']  
  
#отримання налаштованого обїекту к середних  
kmeans = KMeans(n\_clusters=8,  
 init='k-means++',  
 n\_init=10,  
 max\_iter=300,  
 tol=0.0001,  
 verbose=0,  
 random\_state=None,  
 copy\_x=True,  
 algorithm='auto'  
 )  
  
#вчимо прочитаними данними  
kmeans.fit(X)  
  
#Передбачте найближчий кластер, до якого належить кожна вибірка в X  
y\_kmeans = kmeans.predict(X)  
#Формуємо діаграми розсіювання y та x із різним розміром та кольором маркера.  
plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans, s=50, cmap='viridis')  
centers = kmeans.cluster\_centers\_  
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)  
  
#оголошуемо функцію пошуку кластерів  
def find\_clusters(X, n\_clusters, rseed=2):  
 rng = np.random.RandomState(rseed)  
 i = rng.permutation(X.shape[0])[:n\_clusters]  
 centers = X[i]  
 while True:  
 #Ця функція обчислює для кожного рядка в X індекс рядка Y, який є найближчим (відповідно до вказаної відстані).  
 labels = pairwise\_distances\_argmin(X, centers)  
 #обраховуємо середне значення для цетрів  
 new\_centers = np.array([X[labels == i].mean(0) for i in range(n\_clusters)])  
 print(centers)  
 if np.all(centers == new\_centers):  
 break  
 centers = new\_centers  
 return centers, labels  
  
  
centers, labels = find\_clusters(X, 3)  
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  
  
#знаходимо точки центрів для значення минимальної відстані та встановлюємо точки для діагрмми  
centers, labels = find\_clusters(X, 3, rseed=0)  
plt.scatter(X[:, 0], X[:, 1], c = labels, s=50, cmap='viridis')  
  
#Обчислюємо центри кластерів та прогнозуємо індекси кластерів для кожної вибірки  
labels = KMeans(3, random\_state=0).fit\_predict(X)  
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  
  
plt.show()

Chart, scatter chart

Description automatically generated

Рисунок 2. Отримана діаграма розсіювання

З використанням тестових даних ірисів вдалося встановити наступну кластеризацію з декількома центрами зосередження кластерів (рис.2)

Завдання 3: Оцінка кількості кластерів з використанням методу зсуву середнього

*Лістинг програми:*

#Task 3 -------  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import MeanShift, estimate\_bandwidth  
from itertools import cycle

# Завантаження  
X = np.loadtxt('data\_clustering.txt', delimiter=',')

# Оцінка ширини вікна для Х  
bandwidth\_X = estimate\_bandwidth(X, quantile=0.1, n\_samples=len(X))

# Кластеризація даних методом зсуву середнього  
meanshift\_model = MeanShift(bandwidth=bandwidth\_X, bin\_seeding=True)  
meanshift\_model.fit(X)

# Кластеризація даних методом зсуву середнього  
cluster\_centers = meanshift\_model.cluster\_centers\_  
print('\n Centers of clusters: \n', cluster\_centers)  
  
# Оцінка кількості кластерів  
labels = meanshift\_model.labels\_  
num\_clusters = len(np.unique(labels))  
print("\nNumber of clusters in input data =", num\_clusters)

# Відображення на графіку точок та центрів кластерів  
plt.figure()  
markers = 'o\*xvs'  
  
for i, marker in zip(range(num\_clusters), markers):

#Відображення на графіку точок поточного кластеру

#  
 plt.scatter(X[labels==i, 0], X[labels==i, 1], marker=marker, color="black")  
# Відображення на графіку центру кластера  
 cluster\_centers = cluster\_centers[i]  
 print(cluster\_centers[0])  
 plt.plot(  
 cluster\_centers[0],  
 cluster\_centers[1],  
 marker='o',  
 markerfacecolor='black',  
 markeredgecolor='black',  
 markersize=15  
 )  
  
 plt.title("Clasterss")  
 plt.show()

Chart, scatter chart

Description automatically generated

Рисунок 3. Отримана діаграма розсіювання

Після аналізу було отримано наступну діаграму кластерізації вхадних даних. Так як було використано метод зсуву середнього то можна висунути припущення щодо використаного набору. Наприклад що кількість «піків» близька до кількості кластерів.

Завдання 4: Знаходження підгруп на фондовому ринку з викорситанням моделі поширення подібності

Для вирішення цього завдання було переписано код що було наведено в методичних рекомендаціях через те, що запрапонований у ньому функціонал на разі не підтримується в тому вигляді в якому він був описаний. Також через відсутність файлу symbols\_mapping.json дані про компанії були прописані напряму в коді програми.

*Лістинг програми:*

#TAsk 4----  
  
import datetime  
import json  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn import covariance, cluster  
import yfinance as yf  
from yahoofinancials import YahooFinancials  
  
  
  
symbols = ["PLUG", "AAPL", "PFE", "JNJ"]  
names = ["Plug Power Inc", "Apple", "Pfizer Inc", "Johnson & Johnson"]  
  
openList = []  
closeList = []  
  
  
for symbol in symbols:  
 info = yf.Ticker(symbol)  
 start\_date = datetime.datetime(2003, 7, 3)  
 end\_date = datetime.datetime(2007, 5, 4)  
 quotes = info.history(start=start\_date, end=end\_date)  
 openList.append(quotes.Open)  
 closeList.append(quotes.Close)  
  
opening\_quotes = np.array(openList).astype(np.float\_)  
closing\_quotes = np.array(closeList).astype(np.float\_)  
  
quotes\_diff = closing\_quotes - opening\_quotes  
  
X = quotes\_diff.copy().T  
X /= X.std(axis=0)  
  
edge\_model = covariance.GraphicalLassoCV()  
  
with np.errstate(invalid='ignore'):  
 edge\_model.fit(X)  
  
\_, labels = cluster.affinity\_propagation(edge\_model.covariance\_)  
num\_labels = labels.max()  
  
for i in range(num\_labels + 1):  
 print("Cluster", i+1, "==>", ", ".join(names[i]))

Text

Description automatically generated

Рисунок 4. Отримані результат

Для поточної конфігурації з отриманого часового проміжку було отримано два великі кластери, Де компанія Plug потрапила в кластер 1, а інші в кластер 2.

Це можна побачити якщо вивести масив кластерів й побачити мітки [0, 1, 1, 1]

Висновок: